

---

## Lecture6(part2)

Topics covered:  
Memory subsystem

---



# Memory hierarchy

---

- ❑ A big challenge in the design of a computer system is to provide a sufficiently large memory, with a reasonable speed at an affordable cost.
- ❑ Static RAM:
  - ◆ Very fast, but expensive, because a basic SRAM cell has a complex circuit making it impossible to pack a large number of cells onto a single chip.
- ❑ Dynamic RAM:
  - ◆ Simpler basic cell circuit, hence are much less expensive, but significantly slower than SRAMs.
- ❑ Magnetic disks:
  - ◆ Storage provided by DRAMs is higher than SRAMs, but is still less than what is necessary.
  - ◆ Secondary storage such as magnetic disks provide a large amount of storage, but is much slower than DRAMs.

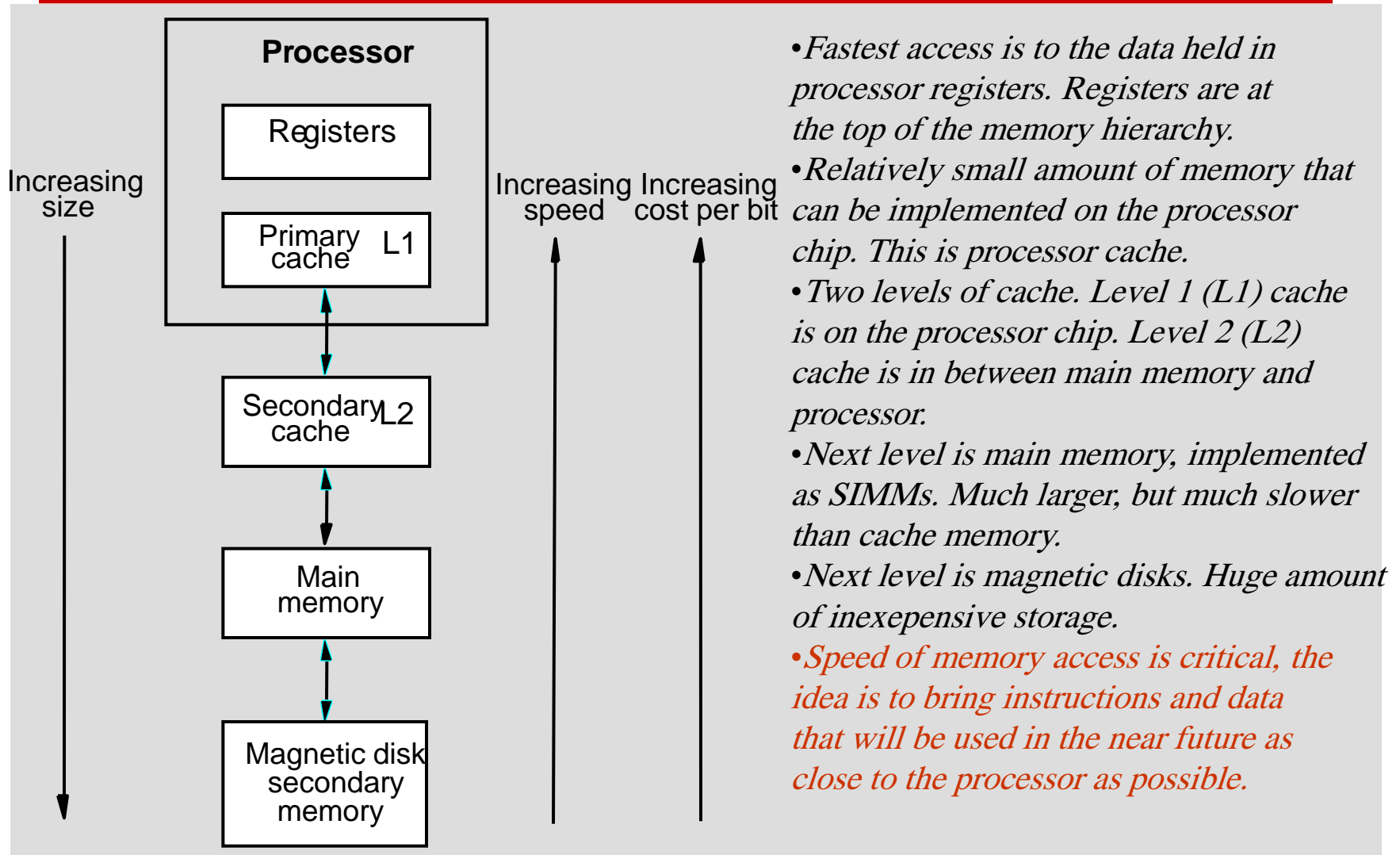


## Memory hierarchy (contd..)

---

- ❑ All these types of memory units are employed effectively in a computer.
  - ◆ SRAM -- Smaller units where speed is critical, namely, cache memories.
  - ◆ DRAM -- Large, yet affordable memory, namely, main memory.
  - ◆ Magnetic disks -- Huge amount of cost-effective storage.
- ❑ Computer memory can be viewed as a hierarchy.

## Memory hierarchy (contd..)





## Cache memories

---

- ❑ Processor is much faster than the main memory.
  - ◆ As a result, the processor has to spend much of its time waiting while instructions and data are being fetched from the main memory.
  - ◆ Major obstacle towards achieving good performance.
- ❑ Speed of the main memory cannot be increased beyond a certain point.
- ❑ Cache memory is an architectural arrangement which makes the main memory appear faster to the processor than it really is.
- ❑ Cache memory is based on the property of computer programs known as "locality of reference".



## Locality of reference

---

- ❑ Analysis of programs indicates that many instructions in localized areas of a program are executed repeatedly during some period of time, while the others are accessed relatively less frequently.
  - ◆ These instructions may be the ones in a loop, nested loop or few procedures calling each other repeatedly.
  - ◆ This is called "locality of reference".
- ❑ Temporal locality of reference:
  - ◆ Recently executed instruction is likely to be executed again very soon.
- ❑ Spatial locality of reference:
  - ◆ Instructions with addresses close to a recently instruction are likely to be executed soon.



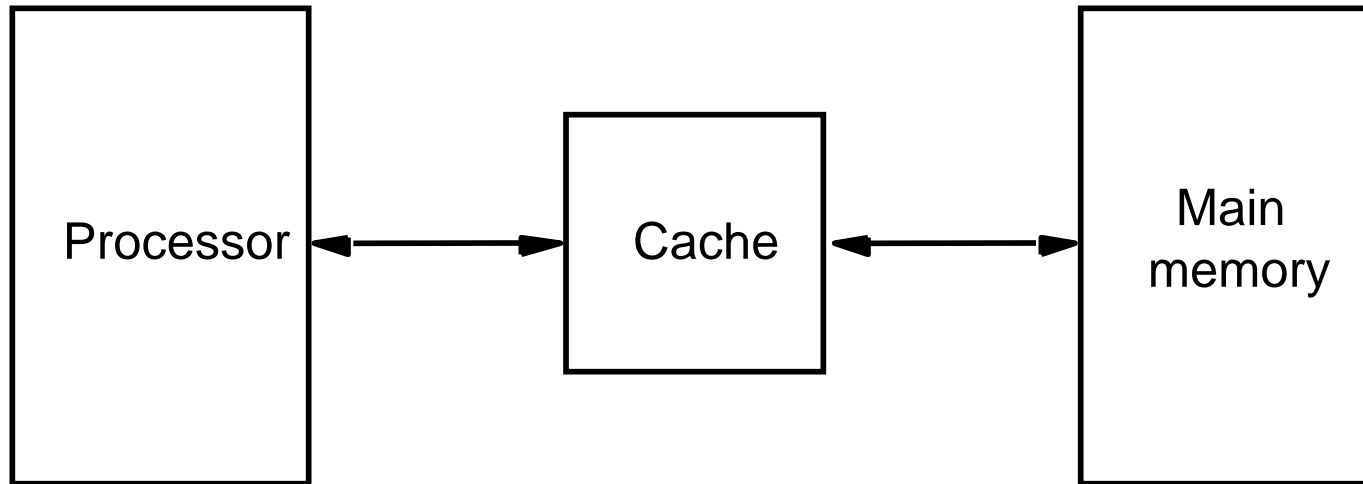
## Locality of reference (contd..)

---

- ❑ Cache memory is based on the concept of locality of reference.
  - ◆ If active segments of a program are placed in a fast cache memory, then the execution time can be reduced.
- ❑ Temporal locality of reference:
  - ◆ Whenever an instruction or data is needed for the first time, it should be brought into a cache. It will hopefully be used again repeatedly.
- ❑ Spatial locality of reference:
  - ◆ Instead of fetching just one item from the main memory to the cache at a time, several items that have addresses adjacent to the item being fetched may be useful.
  - ◆ The term "block" refers to a set of contiguous addresses locations of some size. However, a cache block is called cache line.

# ◆ Cache memories

---



- *Processor issues a Read request, a block of words is transferred from the main memory to the cache, one word at a time.*
- *Subsequent references to the data in this block of words are found in the cache.*
- *At any given time, only some blocks in the main memory are held in the cache. Which blocks in the main memory are in the cache is determined by a “mapping function”.*
- *When the cache is full, and a block of words needs to be transferred from the main memory, some block of words in the cache must be replaced. This is determined by a “replacement algorithm”.*



## Cache memories (contd..)

---

- *Existence of a cache is transparent to the processor. The processor issues Read and Write requests in the same manner.*
- *If the data is in the cache it is called a Read or Write hit.*
- *Read hit:*
  - *The data is obtained from the cache.*
- *Write hit:*
  - *Cache is a replica of the contents of the main memory.*
  - *Contents of the cache and the main memory may be updated simultaneously. This is the write-through protocol.*
  - *Update the contents of the cache, and mark it as updated by setting a bit known as the dirty bit or modified bit. The contents of the main memory are updated when this block is replaced. This is write-back or copy-back protocol.*



## Cache memories (contd..)

---

- *If the data is not present in the cache, then a Read miss or Write miss occurs.*
- *Read miss:*
  - *Block of words containing this requested word is transferred from the memory.*
  - *After the block is transferred, the desired word is forwarded to the processor.*
  - *The desired word may also be forwarded to the processor as soon as it is transferred without waiting for the entire block to be transferred. This is called load-through or early-restart.*
- *Write-miss:*
  - *Write-through protocol is used, then the contents of the main memory are updated directly.*
  - *If write-back protocol is used, the block containing the addressed word is first brought into the cache. The desired word is overwritten with new information.*



## Cache memories (contd..)

---

- *A bit called as “valid bit” is provided for each block.*
- *If the block contains valid data, then the bit is set to 1, else it is 0.*
- *Valid bits are set to 0, when the power is just turned on.*
- *When a block is loaded into the cache for the first time, the valid bit is set to 1.*
  
- *Data transfers between main memory and disk occur directly bypassing the cache.*
- *When the data on a disk changes, the main memory block is also updated.*
- *However, if the data is also resident in the cache, then the valid bit is set to 0.*
  
- *What happens if the data in the disk and main memory changes and the write-back protocol is being used?*
- *In this case, the data in the cache may also have changed and is indicated by the dirty bit.*
- *The copies of the data in the cache, and the main memory are different. This is called the cache coherence problem.*
- *One option is to force a write-back before the main memory is updated from the disk..*

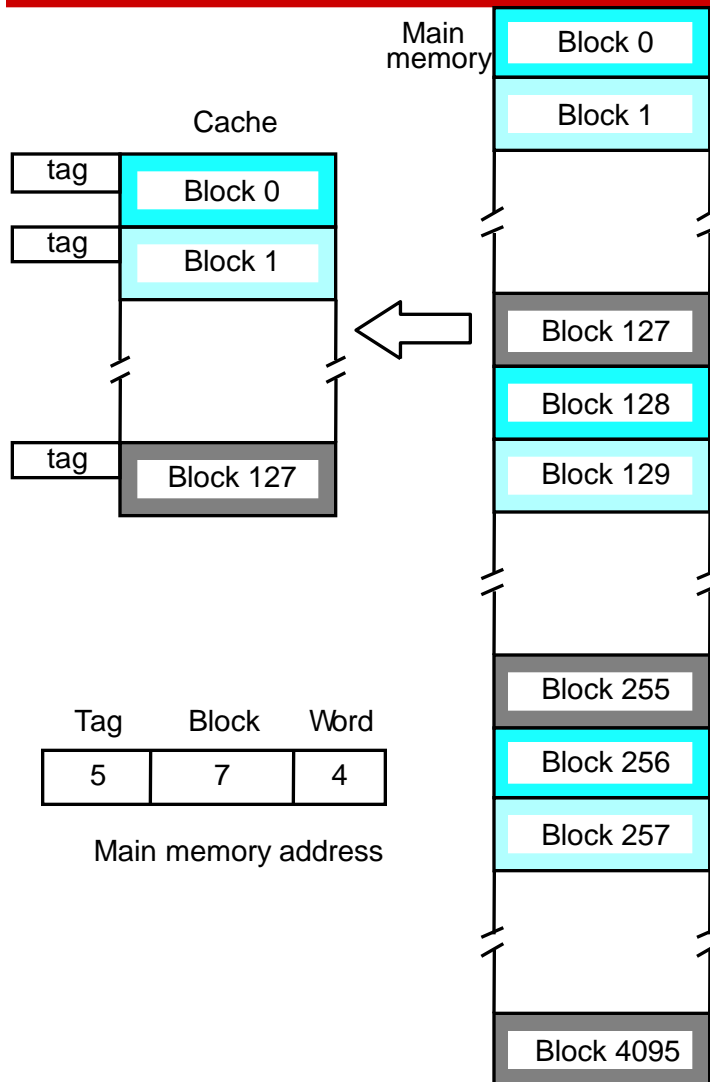


## Mapping functions

---

- ❑ Mapping functions determine how memory blocks are placed in the cache.
- ❑ A simple processor example:
  - ◆ Cache consisting of 128 blocks of 16 words each.
  - ◆ Total size of cache is 2048 (2K) words.
  - ◆ Main memory is addressable by a 16-bit address.
  - ◆ Main memory has 64K words.
  - ◆ Main memory has 4K blocks of 16 words each.
  - ◆ Consecutive addresses refer to consecutive words.
- ❑ Three mapping functions:
  - ◆ Direct mapping
  - ◆ Associative mapping
  - ◆ Set-associative mapping.

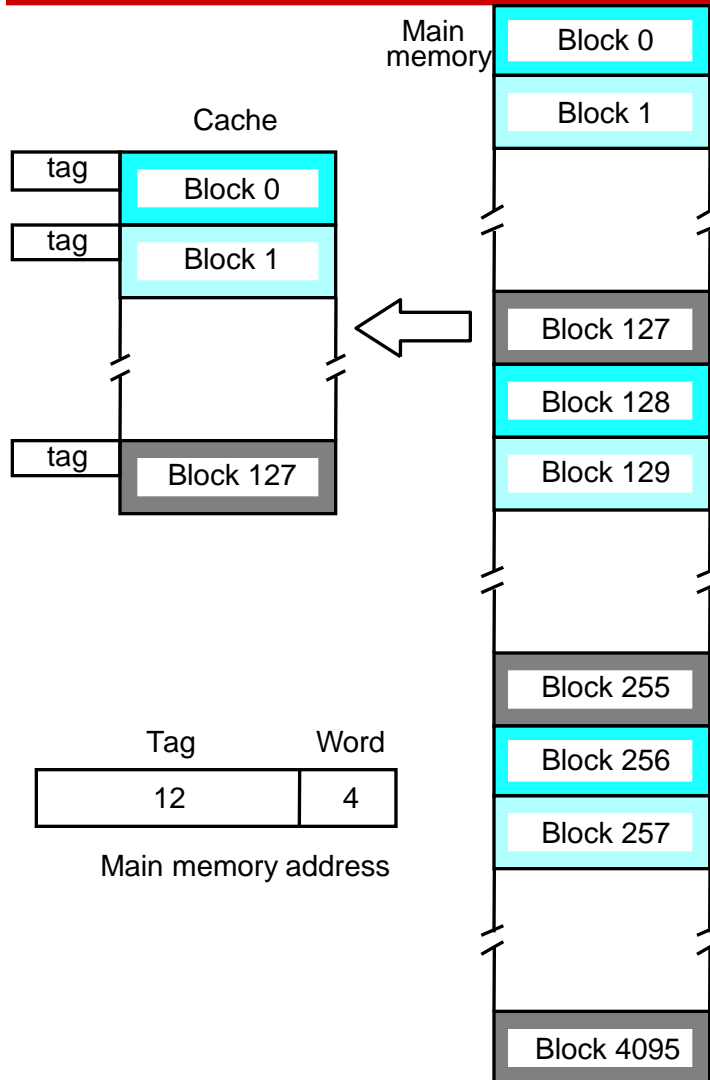
# Direct mapping



- Block  $j$  of the main memory maps to  $j$  modulo 128 of the cache. 0 maps to 0, 129 maps to 1.
- More than one memory block is mapped onto the same position in the cache.
- May lead to contention for cache blocks even if the cache is not full.
- Resolve the contention by allowing new block to replace the old block, leading to a trivial replacement algorithm.
- Memory address is divided into three fields:
  - Low order 4 bits determine one of the 16 words in a block.
  - When a new block is brought into the cache, the next 7 bits determine which cache block this new block is placed in.
  - High order 5 bits determine which of the possible 32 blocks is currently present in the cache. These are tag bits.
- Simple to implement but not very flexible.

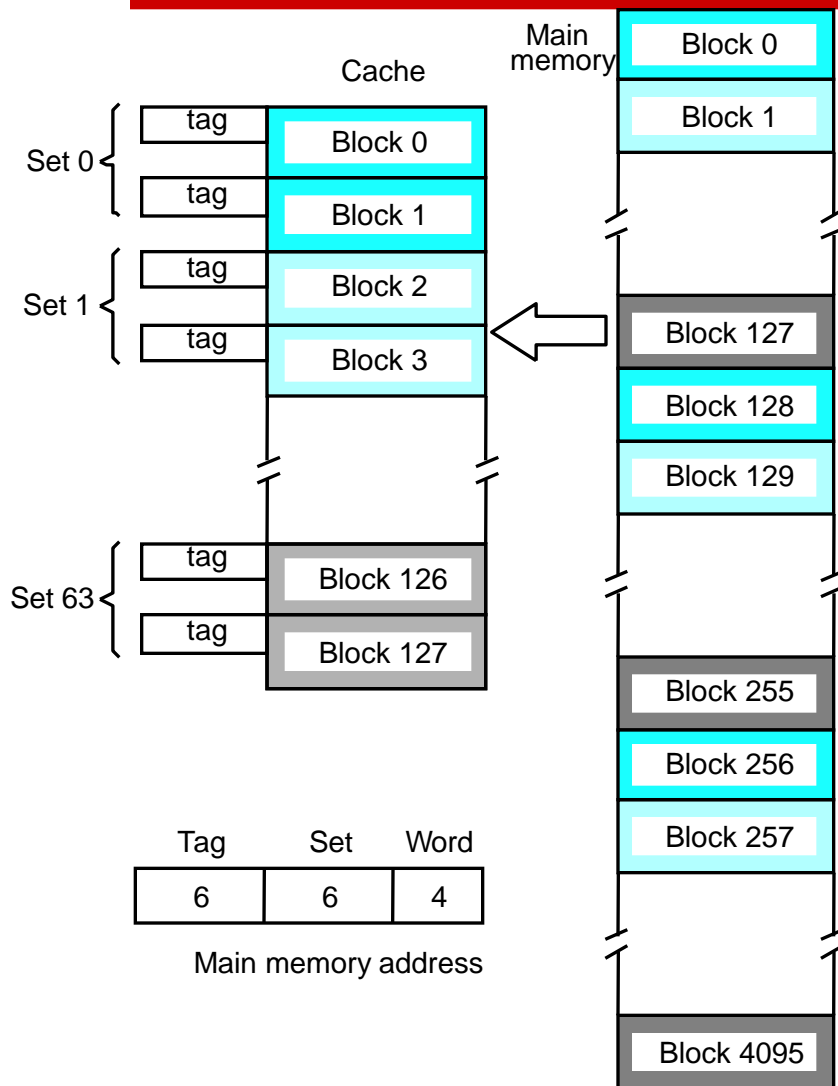


# Associative mapping



- *Main memory block can be placed into any cache position.*
- *Memory address is divided into two fields:*
  - *Low order 4 bits identify the word within a block.*
  - *High order 12 bits or tag bits identify a memory block when it is resident in the cache.*
- *Flexible, and uses cache space efficiently.*
- *Replacement algorithms can be used to replace an existing block in the cache when the cache is full.*
- *Cost is higher than direct-mapped cache because of the need to search all 128 patterns to determine whether a given block is in the cache.*

# Set-Associative mapping



*Blocks of cache are grouped into sets.*

*Mapping function allows a block of the main memory to reside in any block of a specific set.*

*Divide the cache into 64 sets, with two blocks per set. Memory block 0, 64, 128 etc. map to block 0, and they can occupy either of the two positions.*

*Memory address is divided into three fields:*

- 6 bit field determines the set number.
- High order 6 bit fields are compared to the tag fields of the two blocks in a set.

*Set-associative mapping combination of direct and associative mapping.*

*Number of blocks per set is a design parameter.*

- One extreme is to have all the blocks in one set, requiring no set bits (fully associative mapping).
- Other extreme is to have one block per set, is the same as direct mapping.



# Replacement algorithms

---

*Direct-mapped cache, the position that each memory block occupies in the cache is fixed. As a result, the replacement strategy is trivial.*

*Associative and set-associative mapping provide some flexibility in deciding which memory block occupies which cache block.*

*When a new block is to be transferred to the cache, and all the positions it may occupy are full, which block in the cache should be replaced?*

*Locality of reference suggests that it may be okay to replace the block that has gone the longest time without being referenced.*

*This block is called Least Recently Used (LRU) block, and the replacement strategy is called LRU replacement algorithm.*

*LRU algorithm has been used extensively.*

*It provides poor performance in some cases.*

*Performance of the LRU algorithm may be improved by introducing a small amount of randomness into which block might be replaced.*

*Other replacement algorithms include removing the “oldest” block. However, they disregard the locality of reference principle and do not perform as well.*

## ◆ LRU replacement algorithm

---

To use the LRU algorithm, the cache controller must track references to all blocks as computation proceeds. Suppose it is required to track the LRU block of a four-block set in a set-associative cache. A 2-bit counter can be used for each block. When a hit occurs, the counter of the block that is referenced is set to 0. Counters with values originally lower than the referenced one are incremented by one, and all others remain unchanged. When a *miss* occurs and the set is not full, the counter associated with the new block loaded from the main memory is set to 0, and the values of all other counters are increased by one. When a miss occurs and the set is full, the block with the counter value 3 is removed, the new block is put in its place, and its counter is set to 0. The other three block counters are incremented by one. It can be easily verified that the counter values of occupied blocks are always distinct.



## Performance considerations

---

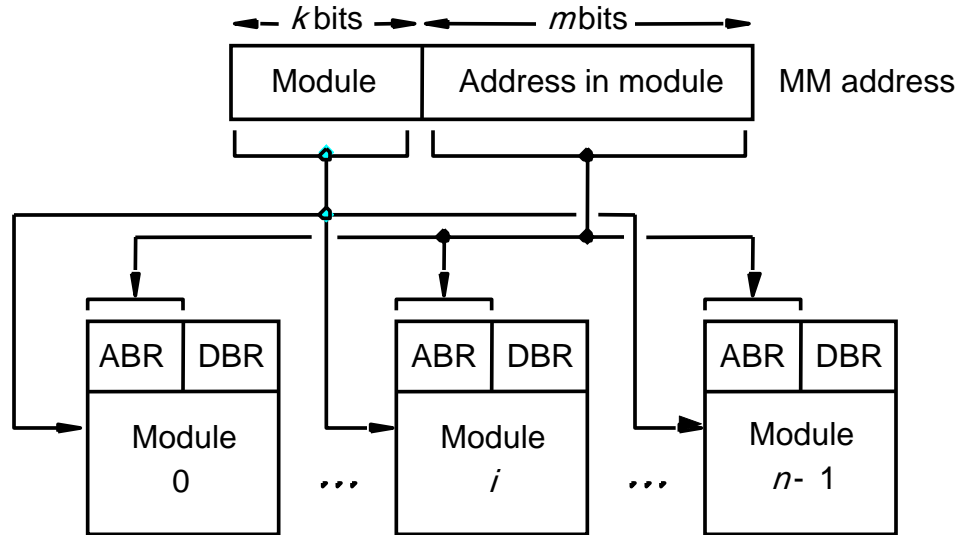
- ❑ A key design objective is to achieve the best possible performance at the lowest possible cost.
  - ◆ Price/performance ratio is a common measure.
- ❑ Performance of a processor depends on:
  - ◆ How fast machine instructions can be brought into the processor for execution.
  - ◆ How fast the instructions can be executed.
- ❑ Memory hierarchy described earlier was created to increase the speed and size of the memory at an affordable cost.
- ❑ Data need to be transferred between various units of this hierarchy as well.
  - ◆ Speed and efficiency of data transfer between these various memory units also impacts the performance.

## Interleaving

---

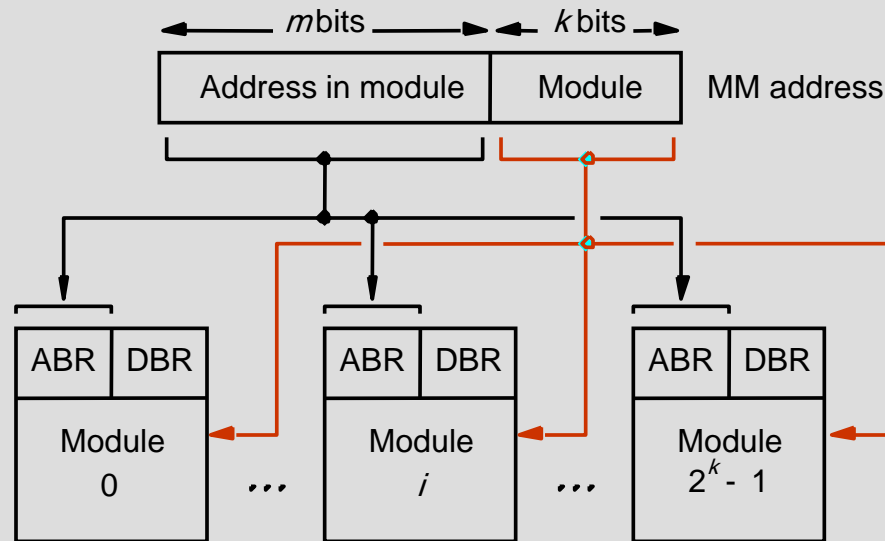
- ❑ Main memory of a computer is structured as a collection of modules.
  - ◆ Each module has its own address buffer register (ABR) and data buffer register (DBR).
- ❑ Memory access operations can proceed in more than one module increasing the rate of transfer of words to and from the main memory.
- ❑ How individual addresses are distributed over the modules is critical in determining how many modules can be kept busy simultaneously.

## ♦ Interleaving (contd..)



- *Consecutive words are placed in a module.*
- *High-order  $k$  bits of a memory address determine the module.*
- *Low-order  $m$  bits of a memory address determine the word within a module.*
- *When a block of words is transferred from main memory to cache, only one module is busy at a time.*
- *Other modules may be involved in data transfers between main memory and disk.*

## ◆ Interleaving (contd..)



*Consecutive words are located in consecutive modules.*

*Low-order  $k$  bits select a module.*

*High-order  $k$  bits select a word location within that module.*

*Consecutive addresses can be located in consecutive modules.*

*While transferring a block of data, several memory modules can be kept busy at the same time.*

*Faster access and high memory utilization.*

*This is called “Memory interleaving”.*



## Memory interleaving Example

---

The effect of interleaving is substantial. Consider the time needed to transfer a block of data from the main memory to the cache when a read miss occurs. Suppose that a cache with 8-word blocks is used, similar to our examples in Section 5.5. On a read miss, the block that contains the desired word must be copied from the memory into the cache. Assume that the hardware has the following properties. It takes one clock cycle to send an address to the main memory. The memory is built with relatively slow DRAM chips that allow the first word to be accessed in 8 cycles, but subsequent words of the block are accessed in 4 clock cycles per word. (Recall from Section 5.2.3 that, when consecutive locations in a DRAM are read from a given row of cells, the row address is decoded only once. Addresses of consecutive columns of the array are then applied to access the desired words, which takes only half the time per access.) Also, one clock cycle is needed to send one word to the cache.



## Memory interleaving Example

---

**Using single memory module:**

$$1 + 8 + (7 \times 4) + 1 = 38 \text{ cycles}$$

**Using four interleaved memory modules:**

$$1 + 8 + 4 + 4 = 17 \text{ cycles}$$



# Performance enhancements

---

## Write buffer

### Write-through:

- *Each write operation involves writing to the main memory.*
- *If the processor has to wait for the write operation to be complete, it slows down the processor.*
- *Processor does not depend on the results of the write operation.*
- *Write buffer can be included for temporary storage of write requests.*
- *Processor places each write request into the buffer and continues execution.*
- *If a subsequent Read request references data which is still in the write buffer, then this data is referenced in the write buffer.*

### Write-back:

- *Block is written back to the main memory when it is replaced.*
- *If the processor waits for this write to complete, before reading the new block, it is slowed down.*
- *Fast write buffer can hold the block to be written, and the new block can be read first.*



# Performance enhancements

---

## Prefetching

- *New data are brought into the processor when they are first needed.*
- *Processor has to wait before the data transfer is complete.*
- *Prefetch the data into the cache before they are actually needed, or a before a Read miss occurs.*
- *Prefetching should occur (hopefully) when the processor is busy executing instructions that do not result in a read miss.*
- *Prefetching can be accomplished through software by including a special instruction in the machine language of the processor.*
  - *Inclusion of prefetch instructions increases the length of the programs.*
- *Prefetching can also be accomplished using hardware:*
  - *Circuitry that attempts to discover patterns in memory references and then prefetches according to this pattern.*



# Performance enhancements

---

## Lockup-Free Cache

- *Prefetching scheme does not work if it stops other accesses to the cache until the prefetch is completed.*
- *A cache of this type is said to be “locked” while it services a miss.*
- *Cache structure which supports multiple outstanding misses is called a lockup free cache.*
- *Since only one miss can be serviced at a time, a lockup free cache must include circuits that keep track of all the outstanding misses.*
- *Special registers may hold the necessary information about these misses.*